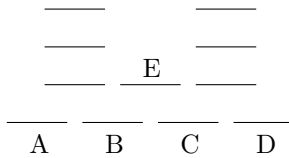
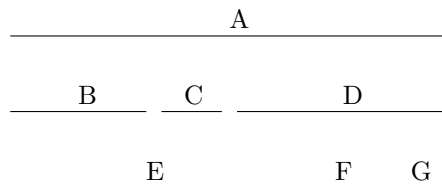


1. This algorithm does not solve the problem of finding a maximum cardinality set of non-overlapping intervals. Consider the following intervals:



Obviously, the optimal solution is $\{A, B, C, D\}$. However, the interval that overlaps with the fewest others is E , and the algorithm will select E first, which precludes it from picking intervals B and C .

2. (a) This algorithm does not solve the interval-coloring problem. Consider the following intervals:



The optimal solution is to put A in one room, $\{B, C, D\}$ in another, and $\{E, F, G\}$ in another, for a total of 3 rooms. However, maximizing the number of classes in the first room results in having $\{B, C, F, G\}$ in one room, and classes A , D , and G each in their own rooms, for a total of 4.

- (b) This algorithm *does* solve the interval-coloring problem. Note that if the greedy algorithm uses N rooms to schedule a given set of classes C , it must be the case that there are N classes that want to meet at the same time. Therefore, N rooms is the fewest number that could be used to schedule C .
3. The greedy algorithm is not optimal for the problem of making change with the minimum number of coins when the denominations are 1, 5, 10, 20, 25, and 50. In order to make 40 Shillings, the greedy algorithm would use three coins of 25, 10, and 5 shillings. The optimal solution is to use two 20-shilling coins.
4. The greedy algorithm is optimal for the problem of making change with the minimum number of coins when the denominations are successive powers of some integer p . We prove this by contradiction. Assume there is some integer D such that the greedy algorithm is not optimal for the set of denominations D^0, D^1, \dots, D^m when making change for x . For $i = 0, \dots, n$,

let g_i denote the number of coins of denomination D^i picked by the greedy algorithm, and let t_i denote the number of denomination D^i picked in an optimal solution. We have:

$$\sum_{i=0}^n g_i D^i = \sum_{i=0}^n t_i D^i = x.$$

Starting from n , let k be the first index for which $g_k \neq t_k$. The greedy algorithm takes as many coins worth D^k as it can, so it must be true that $g_k > t_k$. Furthermore, since both solutions have the same total value, x , the total value of $t_{k-1}, t_{k-2}, \dots, t_0$ must “make up” for the lost value of (at least) one D^k . This can never happen. Note that

$$\sum_{i=0}^{k-1} t_i D^i \leq \sum_{i=0}^{k-1} (D-1) D^i = \sum_{i=0}^{k-1} (D^{i+1} - D^i) = D^k - D^0 = D^k - 1$$

. This says that if $t_0 D^0 + \dots + t_m D^m = x$, then one of the t_i for $i = 0, \dots, k-1$ must be greater than $D-1$. However, if there are ever more than $D-1$ of any denomination of coin (except the largest), then D of those coins can be replaced by one coin of the next largest denomination. This contradicts our assumption that the t_i 's form an optimal solution, since if any t_i differs from g_i , then there must be at least D coins of some smaller denomination among the t_i 's.

5. No solution given.

6. (a) This algorithm is not optimal for the problem of covering points with unit intervals. Let the points to be covered be $A = [-\frac{1}{3}, \frac{1}{4}]$, $B = [\frac{1}{4}, \frac{2}{3}]$, $C = [\frac{2}{3}, \frac{3}{4}]$, $D = [\frac{3}{4}, \frac{5}{3}]$, and $F = [\frac{5}{3}, \frac{1}{3}]$. The algorithm that tries to maximize the number of points covered by the first interval will cover B, C, D, E with the first interval, which forces it to use at least 3 intervals total. The points can, however, be covered with two intervals, $[-\frac{1}{3}, \frac{1}{3}]$, and $[\frac{2}{3}, \frac{5}{3}]$.
- (b) This algorithm is optimal for the problem of covering points with unit intervals. Assume there is a set of points $A = \{a_1, \dots, a_n\}$ such that the solution obtained by the greedy algorithm is not optimal. Call the greedy solution $G = \{g_1, \dots, g_n\}$ and the optimal solution $T = \{t_1, \dots, t_n\}$. Assume the intervals are numbered in increasing order of left endpoint. Starting at the leftmost interval in G , compare G and T . Let k be the number of the first interval for which $g_k \neq t_k$. By the definition of the greedy algorithm, it must be the case that $g_k > t_k$ (meaning that g_k begins further to the right than t_k). Create solution T' by replacing interval t_k with g_k . Since for $i = 1, \dots, k-1$, $g_i = t_i$, solution T' will continue to cover all the points in A . If g_k

overlaps any other interval t_j in T , shift t_j to the right until it no longer overlaps g_k . Continue shifting intervals in T' to the right until there are no more overlaps. Note that T' continues to cover all points in A . By repeating the above process, we can make $T = G$, contradicting our assumption that G is not an optimal solution.

7. The third algorithm is correct. The proof is very similar to the proof of correctness in section 4.3.1 in the text (I also did this proof in class). The structure of the proof is identical, the algebra is just a bit more complicated.
8. (a) This algorithm is incorrect for the problem of minimizing the average difference between the heights of skiers and their skis. Let $p_1 = 5$, $p_2 = 10$, $s_1 = 9$, and $s_2 = 14$. The algorithm would pair p_1 with s_2 and p_2 with s_1 for a total cost of $\frac{1}{2}(1 + 9) = 5$. Pairing p_1 with s_1 and p_2 with s_2 yields a total cost of $\frac{1}{2}(4 + 4) = 4$.
- (b) The algorithm is correct for the problem of minimizing the average difference between the heights of skiers and their skis. The proof is by contradiction. Assume the people and skis are numbered in increasing order by height. If the greedy algorithm is not optimal, then there is some input $p_1, \dots, p_n, s_1, \dots, s_n$ for which it does not produce an optimal solution. Let the optimal solution be $T = \{(p_1, s_{\alpha(1)}), \dots, (p_n, s_{\alpha(n)})\}$, and note the output of the greedy algorithm will be $G = \{(p_1, s_1), \dots, (p_n, s_n)\}$. Beginning with p_1 , compare T and G . Let p_i be the first person who is assigned different skis in G than in T . Let s_j be the pair of skis assigned to p_i in T . Create solution T' by switching the ski assignments of p_i and p_k , where p_k is the person who was assigned s_i in T . Note that by the definition of the greedy algorithm, $s_i \leq s_j$. Also note that by def of $p_i, p_i \leq p_k$. The total cost of T' is given by

$$Cost(T') = Cost(T) - \frac{1}{n}(|p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j|)$$

There are six cases to be considered. For each case, one needs to show that $(|p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j|) \geq 0$.

Case 1: $p_i \leq p_k \leq s_i \leq s_j$.

$$\begin{aligned} |p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j| &= \\ (s_j - p_i) + (s_i - p_k) - (s_i - p_i) - (s_j - p_k) &= 0 \end{aligned}$$

Case 2: $p_i \leq s_i \leq p_k \leq s_j$.

$$|p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j| =$$

$$\begin{aligned} (s_j - p_i) + (p_k - s_i) - (s_i - p_i) - (s_j - p_k) &= \\ 2(p_k - s_i) &\geq 0 \end{aligned}$$

Case 3: $p_i \leq s_i \leq s_j \leq p_k$.

$$\begin{aligned} |p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j| &= \\ (s_j - p_i) + (p_k - s_i) - (s_i - p_i) - (p_k - s_j) &= \\ 2(s_j - s_i) &\geq 0 \end{aligned}$$

Case 4: $s_i \leq s_j \leq p_i \leq p_k$.

$$\begin{aligned} |p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j| &= \\ (p_i - s_j) + (p_k - s_i) - (p_i - s_i) - (p_k - s_j) &= 0 \end{aligned}$$

Case 5: $s_i \leq p_i \leq s_j \leq p_k$.

$$\begin{aligned} |p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j| &= \\ (s_j - p_i) + (p_k - s_i) - (p_i - s_i) - (p_k - s_j) &= \\ 2(s_j - p_i) &\geq 0 \end{aligned}$$

Case 6: $s_i \leq p_i \leq p_k \leq s_j$.

$$\begin{aligned} |p_i - s_j| + |p_k - s_i| - |p_i - s_i| - |p_k - s_j| &= \\ (s_j - p_i) + (p_k - s_i) - (p_i - s_i) - (s_j - p_k) &= \\ 2(p_k - p_i) &\geq 0 \end{aligned}$$

9. This algorithm is correct for the problem of minimizing the total sum of all line penalties. The proof is by contradiction. Assume there is an optimal solution T , and call the output of the greedy algorithm G . Let s_i be the penalty of the i th line of solution S . Let j be the number of the first line in T that is different from the j th line in G . By the definition of the algorithm, $g_i < t_i$. Create a new solution T' by moving the first word of line $i + 1$ in T to the end of line i . Let l be the length of this word. Note that $t'_{i+1} = t_{i+1} + l$ and $t'_i = t_i - l$. Therefore, the the total sum of all line penalties in T' is the same as the total sum of all line penalties of T . T' is more like greedy than T , and has the same total penalty. Contradiction.
10. This algorithm is incorrect for the problem of minimizing the maximum line penalty. Let $L = 5$, and consider the words "AAA", "BB", "CC", and "DDDD". The greedy algorithm produces

AAABB	penalty = 0
CC	penalty = 3
DDDD	penalty = 1

for a maximum line penalty of 3. The optimal solution is

AAA	penalty = 2
BBCC	penalty = 1
DDDD	penalty = 1

for a maximum line penalty of 2.

11. No solution given
- 12.
13. The algorithm is correct for the problem of building an $n \times n$ matrix with zeros and ones such that the sum of all ones in the i th row is r_i and the sum of all ones in the i th column is c_i for all $1 \leq i \leq n$. The proof is by contradiction. Assume there is some input $r_1, \dots, r_n, c_1, \dots, c_n$ for which the greedy algorithm does not give the correct solution. Call any correct matrix T and the matrix generated by the greedy algorithm G . Let i and j be two numbers such that $g_{ij} \neq t_{ij}$. Let $g_{ij} = 1$; this implies that $t_{ij} = 0$. By the definition of the problem, there must be a number $k \neq j$ such that $g_{ik} = 0$ and $t_{ik} = 1$. Create matrix T' by making $t_{ij} = g_{ij}$. T' is not a feasible solution; column j has too many ones and column k has too few. Since the greedy algorithm placed a 1 in g_{ij} and a 0 in g_{ik} , it must be true that $c_j \geq c_k$. Therefore, the number of ones in column k of T' is at most $c - 1$ and the number in column j is exactly $c_j + 1$. There must be at least one number $l \neq i$ such that $g_{lj} = 0, t'_{lj} = 1, g_{lk} = 1$, and $t'_{lk} = 0$. Create a new matrix T'' by making $g_{lj} = t'_{lj}$ and $g_{lk} = t'_{lk}$. Columns j and k now have the correct number of ones. Matrix T'' is now a feasible solution that is closer to G than T . Contradiction.

The case where $g_{ij} = 0$ and $t_{ij} = 1$ is nearly identical.

14. No solution given
15. (a) Since no man can be rejected more than once by a woman, no man can be rejected more than n times altogether. Thus, since at least one man is rejected at each stage (except on the last stage), there can be no more than $O(n^2)$ stages. The bound is asymptotically tight.
- (b) We first note, as a little lemma, that in the proposed courting algorithm a woman always ends up marrying the highest ranked man (according to her preference ranking) who ever showed up on her

porch. Indeed, she will never reject that man and a man will not go away unless rejected :-).

Now, by contradiction, assume that an unstable marriage exists. That is, there is a man x and a woman y such that both x and y prefer each other to their respective partners. By our lemma above, if y prefers x to her current mate, it means that x has never been to her porch. Since x has never been to y 's porch he cannot have been rejected by y . This contradicts the fact that x prefers y to his current mate since he obviously went to his current mate's porch when he could have gone to y 's.

- (c) The proposed courting algorithm *does* produce a man-optimal marriage assignment. Assume by way of contradiction that it doesn't. Then there's an instance where the marriage assignment M given by the algorithm is not man-optimal, so that means some man is rejected at some point by his optimal woman in our algorithm. Consider the first such man, call him x . Let x 's optimal woman (the best possible woman x is married to in any stable matching) be called y_0 . Let the stable matching in which x and y_0 are married be called M' . Since x was rejected by y_0 , a man x_0 prefers to x must have appeared on her doorstep. Call this man x_0 . Since x was the first man rejected by his optimal woman, x_0 has not yet been rejected by his optimal woman, so he likes y_0 at least as much as he likes his optimal woman. But if y_0 prefers x_0 to x and x_0 prefers y_0 to his partner in M' , then M' isn't stable. This contradicts the definition of M' .
- (d) The proposed courting algorithm *does not* produce a woman-optimal marriage assignment. Consider for instance the simple problem with two women W_1 and W_2 , and two men M_1 and M_2 , where the ranking are as follows:

$$\begin{aligned} M_1 &: W_1 W_2 \\ M_2 &: W_2 W_1 \\ W_1 &: M_2 M_1 \\ W_2 &: M_1 M_2 \end{aligned}$$

Since on the first stage M_1 and M_2 go to different women the algorithm immediately terminates and produces the assignment M_1-W_1 and M_2-W_2 . However, this assignment is not woman optimal, since the assignment M_2-W_1, M_1-W_2 is also stable and preferable for both women.

- (e) Not man pessimal. Recycle the instance from the solution to part d.
- (f) The algorithm's solution is woman pessimal. Assume by way of contradiction that it is not woman pessimal for some instance. Call the algorithm's solution on this instance M . Then there is some other

stable matching M' in which some woman y is married to some man x_0 who she likes even less than her husband x in M . So y prefers x to x_0 . And since M is man optimal (by part c), we know x prefers y to his wife in M' . Hence M' is not stable. This contradicts the definition of M' .